Analyzing Optimization Landscape Of Recent Policy Optimization Methods In Deep RL

by

Mahir Asaf Khan
22141075
Adib Ashraf
20241063
Tahmid Adib Amin
22141076

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
May 2022

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Mahir Asaf Khan
22141075

Adib Ashraf
20241063

Tahmid Adib Amin
22141076

Approval

The thesis/project titled "Analysing Optimization Landscape Of Recent Policy Optimization Methods In Deep RL" submitted by

1. Mahir Asaf Khan(22141075)
2. Adib Ashraf(20241063)
3. Tahmid Adib Amin(22141076)

Of May, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 29, 2022.

Examining Committee:

Supervisor:
(Member)

Warida Rashid
Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Riashat Islam
PhD Student
Mila (Quebec AI Institute)
McGill University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

In this work we will analyze control variates and baselines in policy optimization methods in deep reinforcement learning (RL). Recently there has been a lot of progress in policy gradient methods in deep RL, where baselines are typically used for variance reduction. However, there has been recent progress on the mirage of state and state-action dependent baselines in policy gradients. To this end, it is not clear how control variates play a role in the optimization landscape of policy gradients.
This work will dive into understanding the landscape issues of policy optimization, to see whether control variates are only for variance reduction or whether they play a role in smoothing out the optimization landscape. Our work will further investigate the issues of different optimizers used in deep RL experiments, and ablation studies of the interplay of control variates and optimizers in policy gradients from an optimization perspective.

Keywords: Optimization Landscape, Policy Optimization, Deep Reinforcement Learning, Variance Reduction, Control Variates

Acknowledgement

To start off, all praise to the Great Allah (SWT) for keeping us well.
We thank our supervisor Warida Rashid Ma'am for her efforts to guide us and giving us the opportunity to work under her. Lastly, we thank our co-supervisor Riashat Islam for giving us the opportunity to work on a side branch of his PhD topic, giving us tremendous support and enlightenment throughout the whole process, and to our parents for their kind support and prayers.

Table of Contents

List of Figures

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

α	step size of deviation
β	normalized direction vector
Δ_{β}^{J+}	change in objective value with parameters θ_{β}^{+} relative to $J(\theta_0)$
Δ_{β}^{J-}	change in objective value with parameters θ_{β}^{-} relative to $J(\theta_0)$
λ^t	Discount factor at time-step t
$\mathbb{E}_{\pi}$	Expectation over a probability distribution of π
$\pi_{\theta}(s, a)$	state-action policy parameterized by θ
θ	parameter space
θ_0	optimal parameter space
$J(\theta)$	Objective function generated by agent during learning
R_t	Reward received by agent in a state at time-step t
RL	Reinforcement Learning
$V_a(s)$	Adaptive baseline to estimate the baseline for a gradient update

Chapter 1

Introduction

Reinforcement learning is a subset of Machine learning that is expanding at an exponential rate. It consists of software agents who are placed in a controlled environment where they are dealt certain tasks that they need to complete in order to maximize some prioritized rewards. Contrary to Supervised learning algorithms which map function from input to output, RL algorithms tend to only provide the input and do not involve the target outputs. A basic RL algorithm has three elements; the agent (which, in its current state, can choose to carry out a certain action), the environment (reacts to the action done by the agent and provides it with new inputs) and lastly, the reward (cumulative mechanism returned by the environment). In the bigger picture, what most RL algorithms try to achieve is a balance between exploration and exploitation but the immediate goal is to maximize the reward within a certain trial. There are three types of RL implementations: policy-based, value-based and model-based. Policy-based RL involves coming up with a deterministic/stochastic strategy to maximize the reward. Value-based RL attempts to maximize an arbitrary value function. Model-based RL is based on creating a virtual model for a certain environment where the agent learns to perform within the constraints of the environment [1].

To attain an estimation efficiency, some techniques called the Variance Reduction Techniques are used. One such technique is called the Control Variate technique where they assume the simulation objective is to estimate the mean of a random variable X. This method relies on one or more auxiliary random variables called controls and utilizes their known mean to reduce the variance of the estimator for E[X]. Therefore, the CV method can be seen as an approach to extract and transfer information included in controls where information is interpreted broadly [2].

Baseline is a function that does not change the anticipated value when added to an expectation but can significantly affect the variance. Baseline for policy gradient can reduce its high variance while not changing the direction. It takes the actions that are better than the average and increase their probability while decreasing the probability of the actions that are worse than average. This is achieved by calculating the average reward over the trajectory and subtracting it from the reward at the current time step. This type of baseline is called the average reward baseline [3].

1.1 Research Problem

Results of Varying Baselines: From the study on variance reduction techniques done in [4], we have found that baselines can not only impact the optimization process in variance reduction but can also lead to qualitatively different learning curves, even when the variance of the gradients is the same. For example, two different baselines with the same variance can give two different results. The more negative baseline shifts the policy towards a deterministic one by promoting committal behavior whereas the more positive baseline leads to a non-committal behavior, where the policy retains higher entropy for a longer period.

Different baselines can also affect the convergence of natural policy gradient (NPG) such as baselines minimizing the variance can result in a convergence to a deterministic, sub-optimal policy for any positive step size while baselines with larger variance leads to a convergence to the optimal policy. This sort of behavior directly violates the standard assumptions in optimization.

On-policy sampling is a key factor to these convergence issues as it results in a cycle of making bad updates which can lead to worse policies thus resulting in even worse updates.

Committal and Non-Committal behaviors: Imagine we have a sample action ai. If the function r(ai)-b is positive, which is more prominent when the baseline, b is small (more negative), the update rule will increase the probability of taking the action the agent has already taken before regardless of it being correct or not. Because the agent is likely to choose the same actions again, we call this committal behavior.

While a smaller baseline leads to a committal behavior, a larger (more positive) baseline makes the agent question it's own decisions. If the function r(ai)-b is negative,which occurs when b is large, the parameter update decreases the probability of that certain action thus reducing the probability that the agent will retake the action it just took while increasing the probability of other actions. This can slow down the convergence process but it makes it harder for the agent to get stuck. This is called the non-committal behavior.

1.2 Research Objectives

This research aims to compare the results that adding a baseline/control variate has on a reinforcement learning algorithm. Our aim is to figure out the true nature of these with observations as there is still a fog regarding it and their nature of function in reinforcement learning. We are also going to try and find how they affect the optimization landscape of the objective function, as that aspect of reinforcement learning still suffers from reaching the optima and especially so from the curse of dimensionality. The higher the state-action space becomes the more problematic the geometry of the objective function tends to be and it can at times become a chore to try and reach optimal policy without coming across sub-optimal policy and stopping there.

The objectives are:

1. to understand control variates more in-depth
2. to observe their roles in variance reduction
3. to experiment and find out empirically how they aid in the success of an RL task along with if they help in the optimization of exploration

Chapter 2

Related Works

2.1 Policy Gradient Methods

Policy gradient methods are a class of reinforcement learning algorithms which use gradient ascent $\Delta\theta = \alpha\nabla_\theta J(\theta)$, where $\nabla_\theta J(\theta)$ is the policy gradient and α is the learning rate to reach a local maxima of some policy objective function ($J(\theta)$) [5]. There is a lot of literature surrounding policy gradient methods and various methods have been found throughout the research lifetime in this field, such as PPO [6] and TRPO [7] to list a few. Our methodology aims to use the REINFORCE [8] policy gradient algorithm to carry out the objectives of our research as stated previously.

While trust region policy optimization (TRPO) [7] methods offer robustness, they are complex to implement. Proximal policy optimization gets rid of this problem by keeping the robustness and stability while also being simpler to implement [6]. Proximal policy optimization methods regularize the policy updates by having a lower bound to keep policy updates in a certain region [6]. As such, the surrogate objective here is free to be chosen depending on the specific environment. Implementations differ from case to case but the most commonly accepted design choice has been the "clipped objective". This choice has been generally accepted due to the reliable MuJoCo and Atari benchmarks. However, in recent works [9], it is seen that while these choices for policy updates and parameterization work well in territory of current benchmarks, they are prone to failure modes outside of it. Another recent work [10], goes into depth regarding some of these deep policy gradient methods and their algorithms. It has been found that a large portion of the improvement from TRPO to PPO had come about due to small modifications in code [10]. Both of these works delve into well established design choices or algorithms and make us reconsider our approach to deep RL experiments. This has been an inspiration for our work to investigate how the control variates come into play here. Whether the belief of control variates helping in experiments is well put or not.

From [5] we get a general idea about the REINFORCE (Monte-Carlo policy gradient) algorithm and how it works, it is a policy gradient method which utilizes stochastic policy ascent, policy gradient theorem (according to which, $\Delta_\theta J(\theta) = E[\Delta_\theta \log \pi_\theta(s,a) Q^\pi_\theta(s,a)]$, for any differentiable policy $\pi_\theta(s,a)$), and using the return V_t as an unbiased sample of $Q^\pi_\theta(s_t,a_t)$ we get the gradient equation:

$$\Delta\theta_t = \alpha \nabla_\theta \log \pi_\theta(s, a) v_t \tag{2.1}$$

We use this in the algorithm of Reinforce: we initialize the parameters θ arbitrarily, then, we loop for each episode distributed over the policy π_θ, in which we loop over every time step $t = 1$ until $T - 1$ where we update the value of θ with the equation:

$$\theta_{next\,time\,step} = \theta_{current\,time\,step} + \Delta\theta_t \tag{2.2}$$

In the above idea we can think of $\Delta\theta_t$ as the reward increment, α as the non-negative factor, $\nabla_\theta \log \pi_\theta(s, a)$ as the offset reinforcement and v_t as characteristic eligibility, the description of a REINFORCE algorithm [8]. Considering the simplicity of the algorithm and the problems that accompany it which also accompany other policy gradient methods (such as high variance, optimization of exploration) this is currently considered a good choice for the implementation within our research problem.

2.2 Control Variates and Baselines

Control variates are a variance reduction technique used in Monte Carlo methods. The main idea is such that for a function (say, F), we have another function (say, G), such that $F \approx G$, to reduce its variance during sampling we calculate F in such a manner that it's: sampled $F - sampled\, G + G$, as the variance of the above formula is less than the variance of F, we will get a reduced variance by using that. Here G can be considered the control variate [11]. A popular implementation of control variates can be considered to be Baselines [4], [8], [12].

By now, it is common knowledge that baselines do not change the anticipated results when used on a reinforcement learning algorithm but can affect the variance. While this holds true for most cases, a research done by [8] proved otherwise. According to their paper, baselines in fact do have an impact on the optimization process beyond variance reduction and lead to qualitatively different learning curves, even when the variance of the gradients is the same. They mention the different ways baselines can affect the results such as negative baselines leading to a committal behaviour and positive baselines ending in a non-committal one; baselines minimizing the variance can result in a deterministic, sub-optimal policy while baselines with a large variance can lead to an optimal policy; on-policy sampling that can lead to cycle of making bad updates leading to worse policies and many more.

They proceed to experiment with different baselines to find solutions for the mentioned problems. Committal behaviours that lead to the agent choosing the same action over and over can be solved by picking a baseline with a lower variance that leads to both possible updates to be equal resulting in the agent being equally likely to pick any two of the given options. On-policy sampling is one of the root causes

of committal behaviours as it updates at each step thus changing the policy which in turn affects the distribution of rewards obtained. A simple solution to avoid this issue according to the authors is to sample actions from a behaviour policy that selects every action with sufficiently high probability. Such a policy would make it impossible to choose the same, sub-optimal action forever.

Some other methods to solve these convergence issues can be:

- Reducing the stepsizes that might result in the policy not converging as quickly towards a sub-optimal deterministic policy
- Adding entropy regularization to the objective which would prevent the policy from turning into a deterministic one
- Introducing bias in the updates
- Adding exploration policies and many more.

2.3 Policy Optimization

Albeit policy based learning is an optimization problem [5], it is not exempt from the problems mentioned previously. One such problem that arises is the issue of Objective function exploration. In [13] the authors conducted research on the implementation of baselines and their role in optimization. While they did show that standard rules of stochastic optimization are violated for Reinforcement Learning problems, the belief that variance reduction is the only gain from baseline implementation and that they also significantly effect optimization, they observed the effect on optimization could not be explained in terms of the variance reduction, another weird phenomena observed was that lower variance could prove to be deleterious towards reaching an optimal solution. However they did not show the effect of baselines and variance on the optimization landscape itself which might provide a probable answer to the relationship between variance reduction and optimization.

2.4 Landscape Analysis

In [14] it can be observed that the authors conducted an analysis on the idea of policy optimization based on exploration of the objective function. In their research they have provided an effective tool to interpolate the objective function landscape through combining Linear Interpolations with the generation of Objective function geometry using external deviations. The idea of the second bit was such that for an object function J around an optimal parameter space θ_0 would first sample directions β uniformly on a unit ball and then would investigate how J is changing by calculating a set of forward and backward parameters and their corresponding Objective function values of J. From there they also calculated the change in the surrounding landscape around the optimal parameter θ_0, this gave them an idea surround the nature of of the landscape relative to the optimal parameter space. They then decoupled, of the surrounding landscape, the information about the gradient and the curvature to observe the changes in them and their nature around in the

directions d as sampled before. A conclusion of this research was the suggestion that smoothening of the objective function could enable better learning by better enabling exploration.

Chapter 3

Work Plan

Our research tries to not only understand the role of a control variate in reinforcement learning but also to see the difference in optimization landscape with and without the use of a control variate. To do all these, we need to take a RL algorithm and run some simple tasks on it with and without a baseline/control variate and plot the optimization landscape. Figure 1 gives a higher level overview of the workflow.

(a) Start: Starting the thesis.

(b) Take reinforce algorithm.

(c) Running it on some simple tasks using OpenAI Gym (such as CartPole). [15]

(d) See how to plot the optimization landscapes.

(e) Running the experiments with and without baseline/control variate.

(f) Observe the differences between the optimization landscape WITH and WITHOUT baselines.

(g) Is it possible to draw any conclusion from it - ie, can we say whether the landscapes are smoother, and avoids local minima when we add a control variate?

(h) If the landscape is affected we will make notes and observations as to how they are affected along with the results of effect of it on variance.

(i) If the landscape is not affected we will make notes as to how the variance affected and the nature of control variates and baselines.

(j) End: Completion of Undergraduate Thesis.

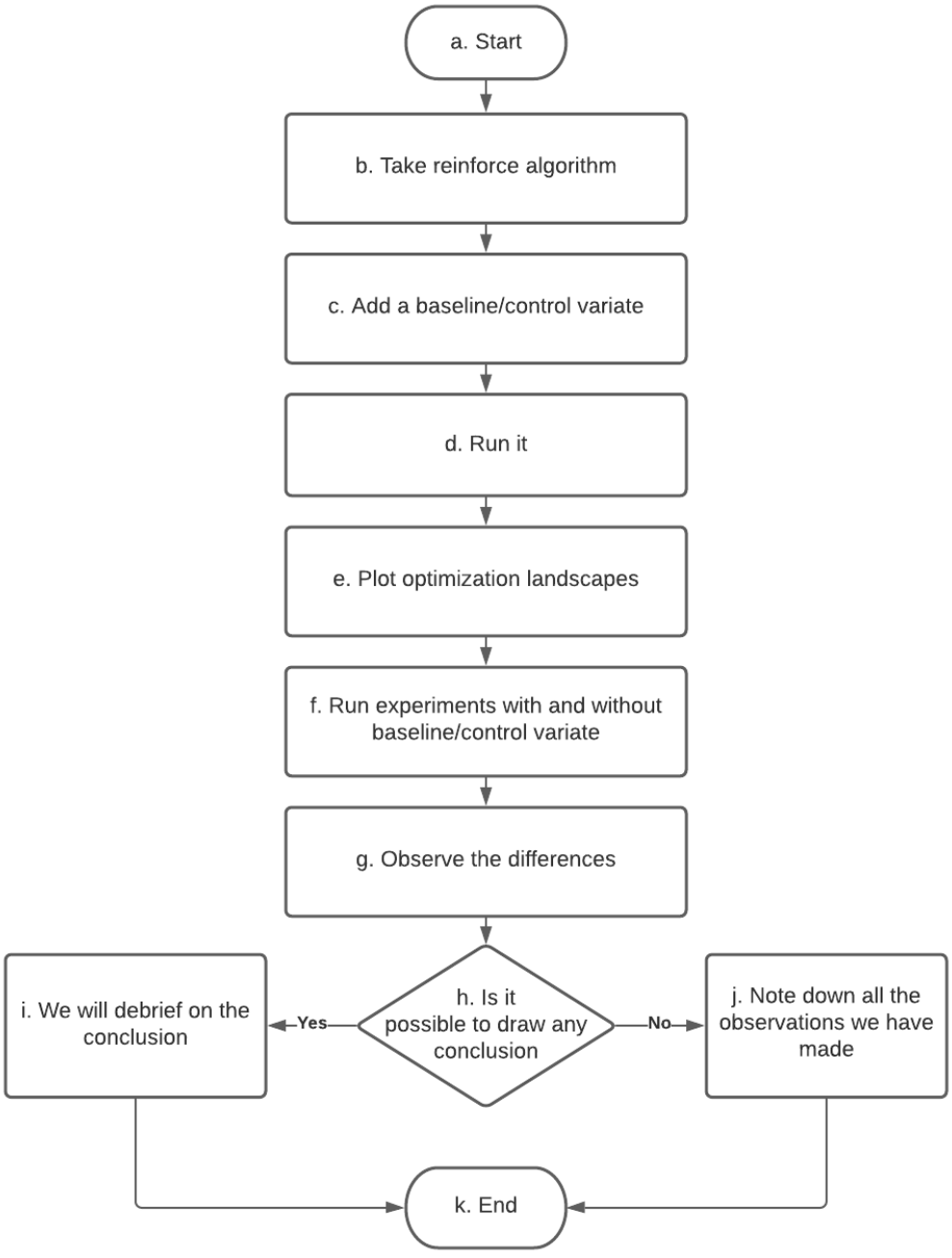


Figure 3.1: Flowchart of the thesis work plan

Chapter 4

The Reinforce Algorithm

REINFORCE [8] proposed by Ronald J. Williams as a general solution of associative reinforcement learning problems implementable through a setup of Perceptron Units (essentially Neural Networks). It is a very straightforward policy gradient algorithm which uses gradient ascent by adding $\Delta\theta_t$ which contains the policy gradient $(\nabla_\theta \log_\theta(s,a)v_t)$ which helps the algorithm to fit to and traverse a high dimensional policy optimization landscape which enables us to achieve an optima for the task it is being implemented on. The implementation of the Reinforce algorithm is as such:

1. **INITIATE** a differentiable policy $\pi_\theta(s,a)$ for the given task
2. **INITIATE** policy parameters θ for the task
3. **DEFINE** a value for the learning rate (α), $\alpha > 0$
4. **FOR** each episode e through n episodes:
   Generate an episode of the task with a set of states $(Ss_1, s_2, \ldots s_T)$, actions $(Aa_1, a_2, \ldots a_T)$,, and rewards $(Rr_1, r_2, \ldots r_T)$ using $\pi_\theta(s,a)$

   FOR $t = 1, t < T$

   $$\theta_{next\,time\,step} = \theta_{current\,time\,step} + \Delta\theta_t \ (where\ \Delta\theta_t = \alpha\nabla_\theta \log \pi_\theta(s,a)v_t)$$

Chapter 5

Adding A Baseline

As previously mentioned Baselines are a form of control variates [4], [8], [12]. It is an excellent method of overcoming the struggle of the bias-variance tradeoff as while implementing it, it doesn't add any additional bias to the model [16]. A study conducted by John Schulman in 2015 [17] showed further that adding a Baseline yields the lowest possible variance, with the catch being that the baseline function must also be sampled.

When adding a baseline to REINFORCE there is a subtle change in the implementation such that the following portion of the implementation presented in **section 4** is changed, where the newly added portion is at the end:

$$\theta_{next\,time\,step} = \theta_{current\,time\,step} + \Delta\theta_t \;(where\,\Delta\theta_t = \alpha\nabla_\theta \log \pi_\theta(s,a)(v_t - baseline(s_t)) \tag{5.1}$$

Where in the above equation, $baseline(s_t)$ represents the baseline function. We selected from [16] a simple implementation which is followed by [18] for our experiments to run experiments mentioned in Chapter 4 and 5, $V_a(s)$ as the baseline to criticize trajectories taken by the learning algorithm. It is in essence an Actor Critic RL algorithm where the baseline acts as a critic, hence an implementation without the baseline is an RL algorithm with only an actor. This gives us the ability to have insight as to how the addition of baseline truly effects policy gradient methods.

Chapter 6

Landscape Analysis Methodology

In section 2.4 we discussed how the authors in [14] scanned the cost function landscape to attain the idea of its nature surrounding the optimal parameter cost function value. In their work they analyzed the change in Objective landscape brought about by Entropy Regularization. For the purposes of our research we will be using the same methodology to check the degree of change brought about by Baselines instead. The 'cost' function in case of the class of learning algorithms that is Reinforcement Learning is the expected discounted sum of rewards given by:

$$J(\theta) = \mathbb{E}_{\pi}[\sum_{t=1}^{\infty} \lambda^t R_t \ ] \tag{6.1}$$

The aim of the learning agent is to maximize $J(\theta)$, making this a maximization problem and making the objective landscape concave in nature. The purpose of the discount factor is to make the agent focus more on the immediate rewards rather than the ones earned previously.
After reaching the optimal π(policy) parameterized by θ_0 we deviate from the parameter space and initiate landscape analysis. We apply the deviations to the parameters using α in the direction d, where $\beta = \frac{\beta\prime}{||\beta\prime||}$ where $\beta\prime \sim N(0,1)$ and $\beta \in \mathbb{R}^p$ where p is equal to the amount of parameters in the parameter space of the policy estimator neural network from [18] as it is the neural net parameterizing our policy. Then we calculate the parameters surrounding the given parameter space with:

$$\theta_{\beta}^{+} = \theta_0 + \alpha\beta \tag{6.2}$$

$$\theta_{\beta}^{-} = \theta_0 - \alpha\beta \tag{6.3}$$

The equation in 6.2 and 6.3 gives us the set of forward and backward set of parameter spaces around θ_0. We use these to compute $J(\theta_{\beta}^{+})$ and $J(\theta_{\beta}^{-})$ which in turn allows us to find the change in objective due to the change in parameters for the forward and backward points using relative to the optimal parameters using:

$$\Delta_{\beta}^{J+} = J(\theta_{\beta}^{+}) - J(\theta) \tag{6.4}$$

$$\Delta_{\beta}^{J-} = J(\theta_{\beta}^{-}) - J(\theta) \tag{6.5}$$

We then scan the landscape for increasing values of step size in the direction of d. Alongside the stated conditions and combinations of Δ_β^{J+} and Δ_β^{J-} in the aforementioned literature [14] we also find two peculiar states of these metrics, such that:

$$\Delta_\beta^{J-} < 0 \, and \, \Delta_\beta^{J+} \approx 0 \tag{6.6}$$

$$\Delta_\beta^{J+} < 0 \, and \, \Delta_\beta^{J-} \approx 0 \tag{6.7}$$

Our intuition for the geometrical conditions in both 6.6 and 6.7 is that the curve is in transition from having a negative slope and going for some maxima that is a plateau, since for ,relative to the reached optima, there is a negative change in one direction there is 0 change in the other.
As they are, Δ_β^{J+} and Δ_β^{J-} have information about gradient and curvature in tandem, in [18]. To decouple them the authors used the following formulae:

$$\Delta_\beta^{J+} + \Delta_\beta^{J-} \tag{6.8}$$

$$\Delta_\beta^{J+} - \Delta_\beta^{J-} \tag{6.9}$$

Expression 6.8 gives us information about the eigenvalue spectrum, hence information about the curvature and expression 6.9 gives us information about the gradient surrounding surrounding $\mathrm{J}(\theta_0)$. To implement this they took $((\Delta_\beta^{J+}, \Delta_\beta^{J-}))$ and projected them on the diagonal then plot histograms from the resulting projection values.
We select the seed that has the most apparent difference for the curves of rewards over a number of episodes (decided by the difference between the no. of episodes needed to reach optimality and the shape of the curve, as they have the highest difference in learning dynamics) and select the optimal parameters of its policy estimator neural network.
The code for the above mentioned steps has been taken from the research done in [14] and modified for the implementation[18] we have used for policy gradients and their baseline implemented counterpart.
We also modified the code in [18] for providing a function to estimate the value of 6.1. Using the function to estimate $\mathrm{J}(\theta_0)$ and the methodology explained in this chapter previously we capture the nature of the landscape surrounding $\mathrm{J}(\theta_0)$ relative to it. By analyzing the nature of the landscape local to the optimal parameters θ_0 and capturing information about the gradient and the curvature surrounding it we can observe and understand how adding a baseline to basic policy gradients (the reinforce algorithm) can affect the algorithms objective function landscape and get intuition as to how the exploration of the agent leads to its result. Experimentation and results are displayed to be analyzed in the next chapter.

Chapter 7

Running REINFORCE with and without Baseline on simple tasks

Unlike Supervised and Unsupervised learning, Reinforcement Learning is a Machine Learning Methodology that uses experience to understand how to adapt to an environment successfully. We will be running the Reinforce Algorithm with and without baseline and try to analyze the results on these two tasks, CartPole [19] (a classic control task) and LunarLander [20] (a Box2D task). We have chosen implementations on these tasks that have used Neural Networks as the Policy and Baseline function approximators, as Reinforce was proposed with the use of Neural Networks in mind [8], and also because of the fact that they follow the law of Universal Approximation [21].

7.1 CartPole

The CartPole problem (also known as the Inverted Pendulum problem) is basically an environment consisting of a cart with a pole attached to it at its center using a single joint. The pole begins at an upright position but can fall left or right due to gravity. The job of the agent is to apply a force of +1 or -1 on the cart to move it right or left respectively on a frictionless track so as to keep the pole upright. Based on how long the agent can balance the pole, it will receive a reward of +1 for each timestep. An episode will terminate if the pole tilts more than 15 degrees from its vertical position and/or the cart moves more than 2.4 units from its central position. [19]

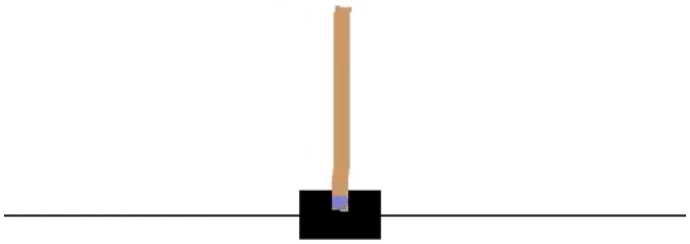


Figure 7.1: Ideal state of the cartpole

7.1.1 Code implementation related to CartPole

For experimenting on Reinforce over the Cartpole environment, we used the implementation written by [18]. This implementation provides the code for both implementing Reinforce with and without a baseline. The implementation with and without baseline is not too different in code. For baseline, another neural network is created for the value function and the baseline loss is calculated in the Agent class, under the 'train' method. Additionally, the actor loss and value function loss is also calculated there. The need for this is explained in section 5, "Adding a Baseline".

7.1.2 Results through CartPole

The hyperparameters setup of this implementation is: number_episodes = 1000, actor learning rate = 0.002, value function learning rate = 0.002. We ran the test 5 times with different seeds. To observe how the model performs on average we ran the model 5 times, for both with and without baselines. The results can be found in Figure 7.2 and Figure 7.3.

In the graphs the orange lines represent the data-points plotted, and the blue lines represent a smoothed running average of 10 episodes. On average it seems that Policy Estimation with a baseline reaches optimal solution faster and there can be seen more drastic dips for the estimation without baseline. The tests without baseline also took longer on average to end. If we look at the raw iterations themselves it can be observed that for iterations using baseline the algorithm reached the optimal solution of 200 points as early as less than 300 episodes into runtime and also being able to retain it, enabling an early stop. There can also be seen more variance for the tests without baseline.

7.1.3 Observations for landscape analysis for cartpole

For cartpole we take 4000 samples around the optimal parameters in direction β for both reinforce policy gradient without and with baseline function.

As it can be observed in 7.5 even for very large step-sizes (until step size = 30), $\Delta_\beta^{J+} \approx 0$ and $\Delta_\beta^{J-} \approx 0$, meaning according to [14], almost all the points lie on an almost flat region indicating that the curve is in such a position that is essentially a plateau. In comparison to that in 7.5 we can see minuscule deviations at a step size as early as 10, for when a baseline is implemented to the reinforce gradient update. For step sizes larger than 30 we can see there is significant deviation of the points relative to the optima. From what can be seen at step size 30 . For implementation without baseline in 7.4 however it begins to deviate from step size = 10 and deviates greatly throughout the rest of the step sizes. From plotting the histograms about gradient and curvature we can in 7.7 and 7.8 and comparing them with their baseline added update counterparts in 7.9 we can see that the gradient and the eigenvalue spectra is much more scattered for the implementation without the baseline update than the one for with baseline update indicating that relative to the optima at θ the local landscape around that point still has its curvature and gradient value closely bounded near the optimal parameters even further away for

an added baseline, whereas for no baseline it deviates heavily as we go further away introducing steep and unsafe valleys on the objective function landscape making it harder to converge which would explain the difference in learning curves for same seed setup.

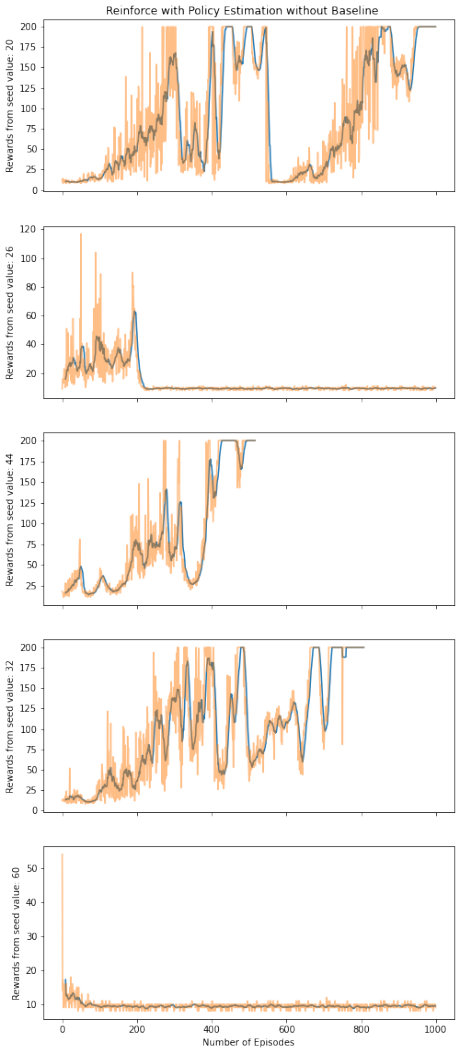


Figure 7.2: The result with 5 different seeds without baseline

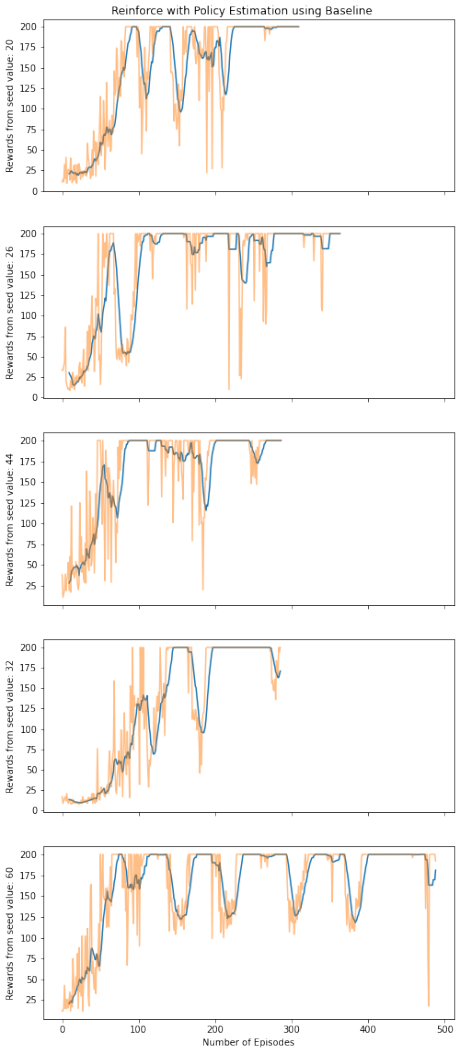


Figure 7.3: The result with 5 different seeds with baseline

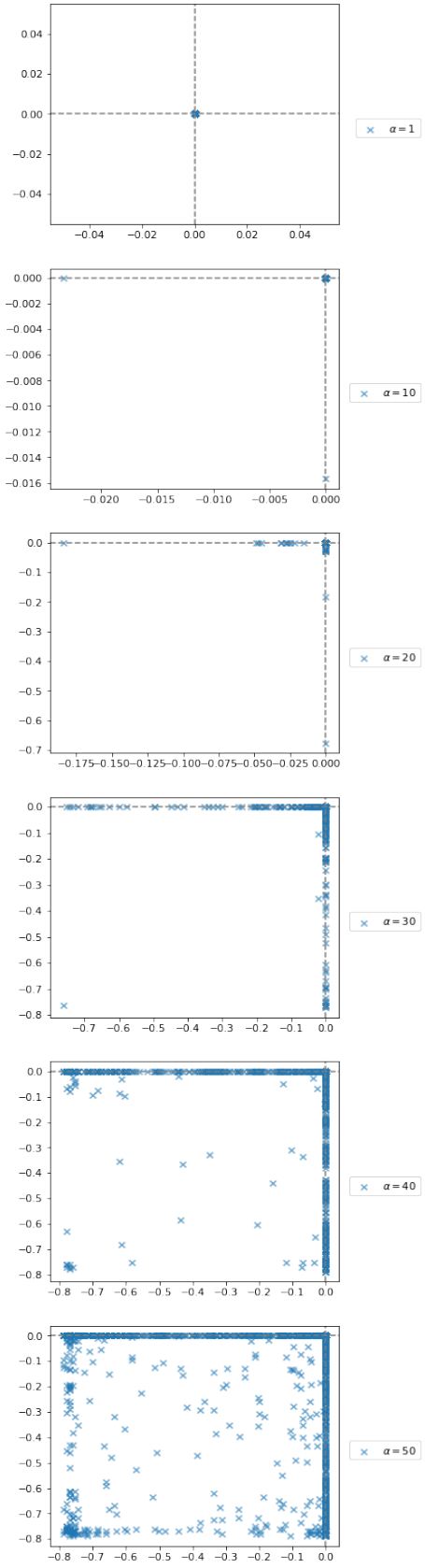


Figure 7.4: Cartpole Landscape without baseline (x-axis: Δ_β^{J+}, y-axis: Δ_β^{J-})

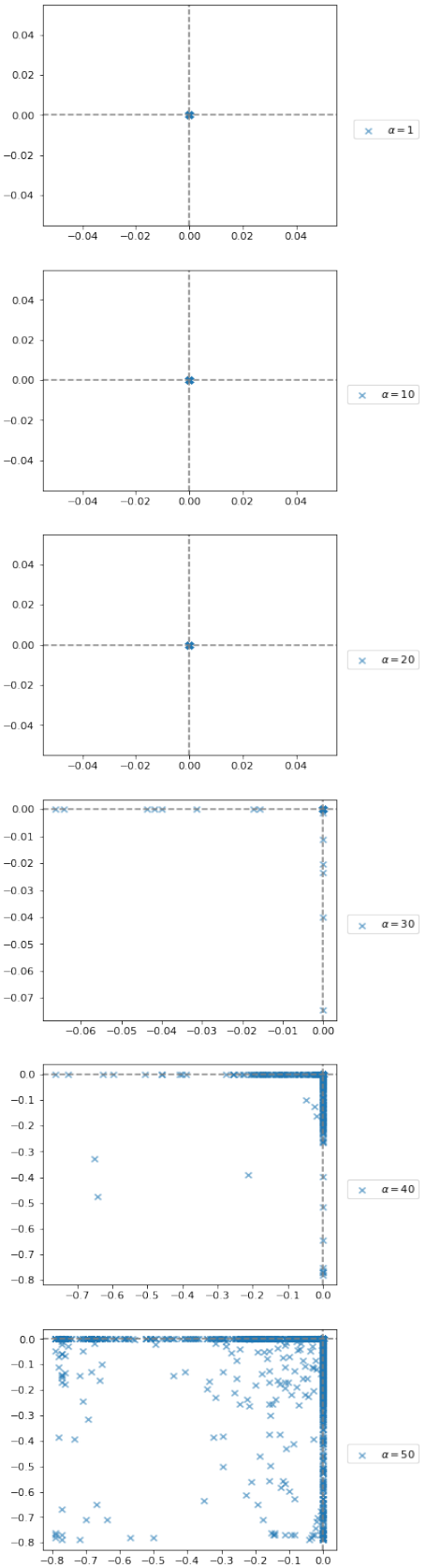


Figure 7.5: Cartpole Landscape with baseline (x-axis: Δ_β^{J+}, y-axis: Δ_β^{J-})

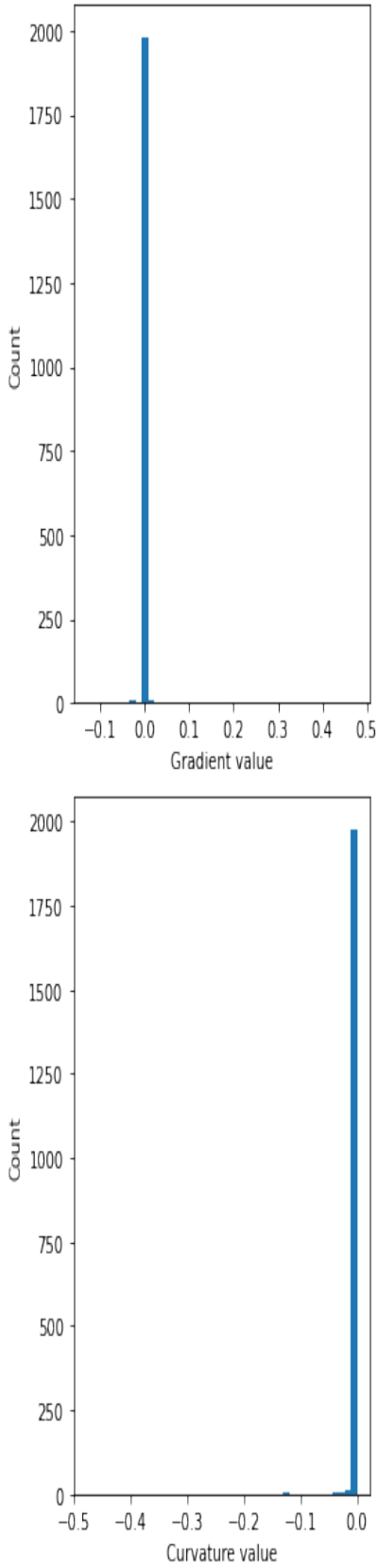


Figure 7.6: Cartpole Histogram Step 20 without baseline

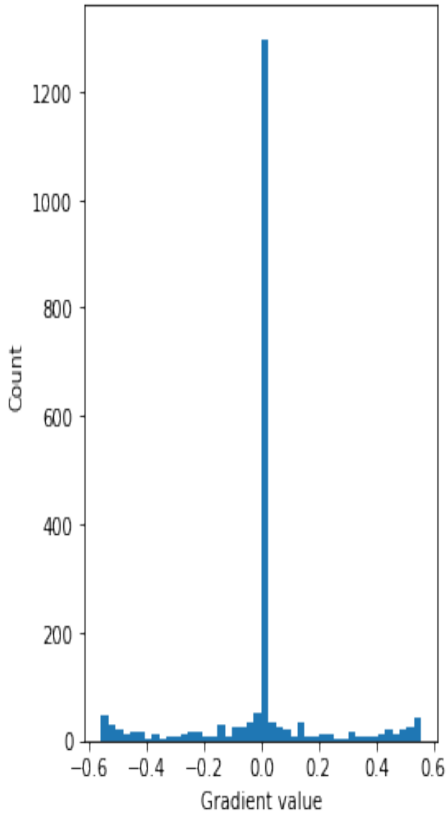


Figure 7.7: Cartpole Histogram Step 40 without baseline

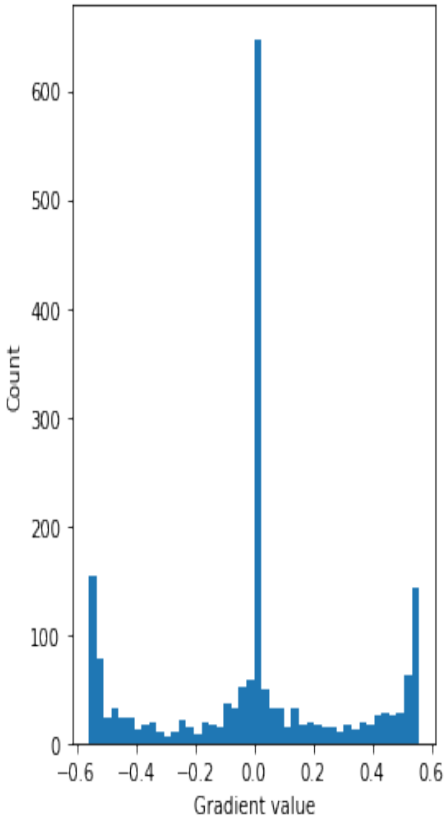
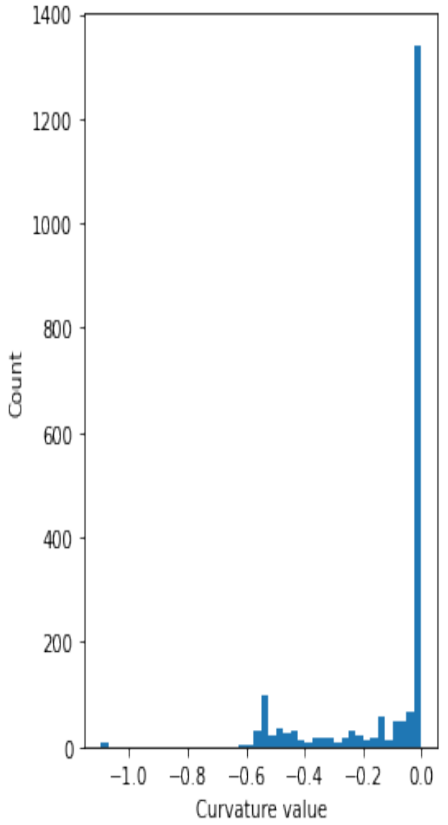
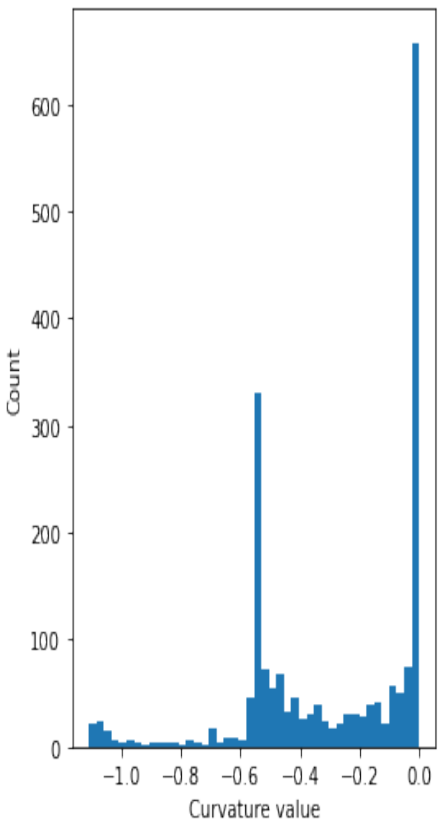


Figure 7.8: Cartpole Histogram Step 50 without baseline





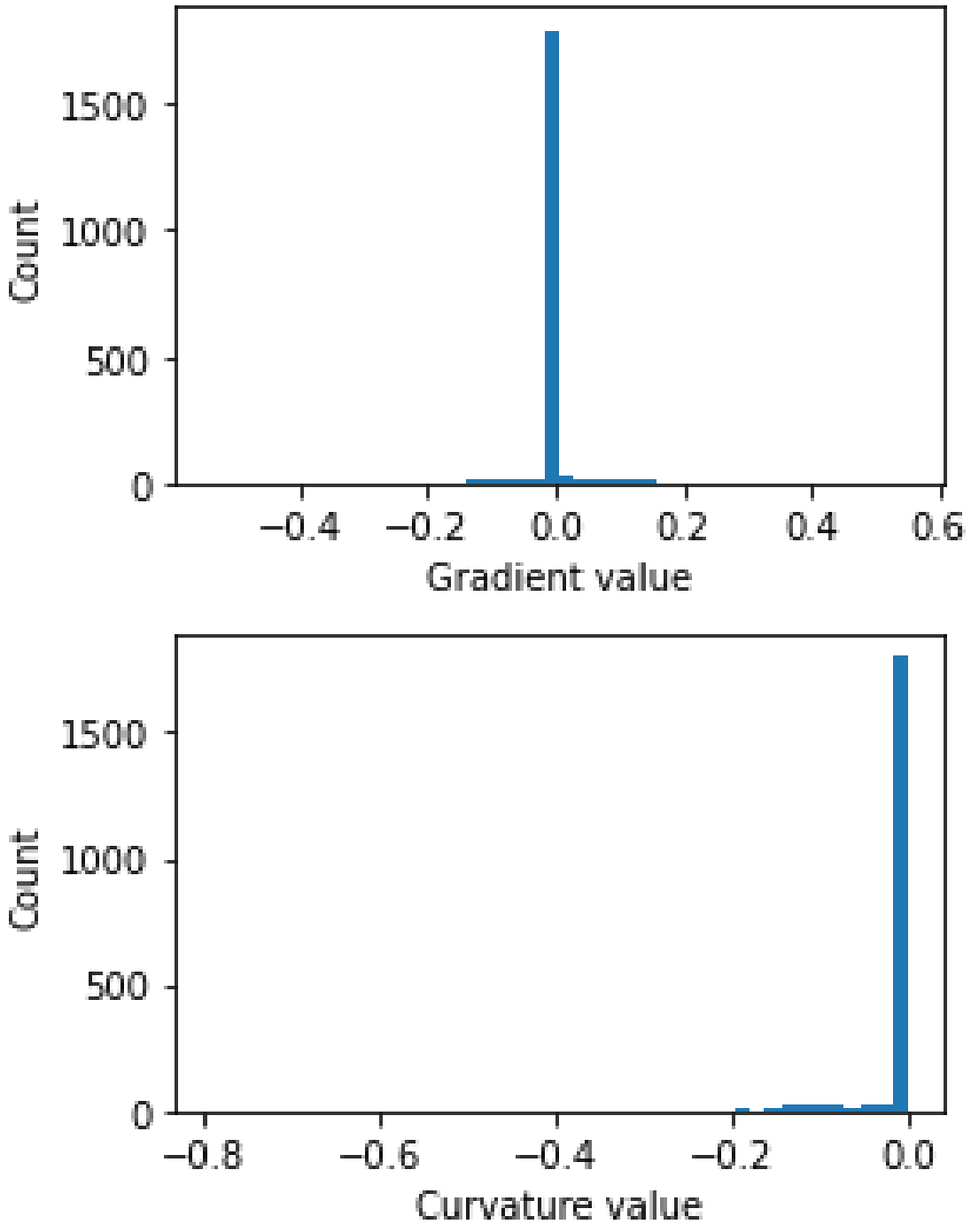


Figure 7.9: Cartpole Histogram Step 40 with baseline

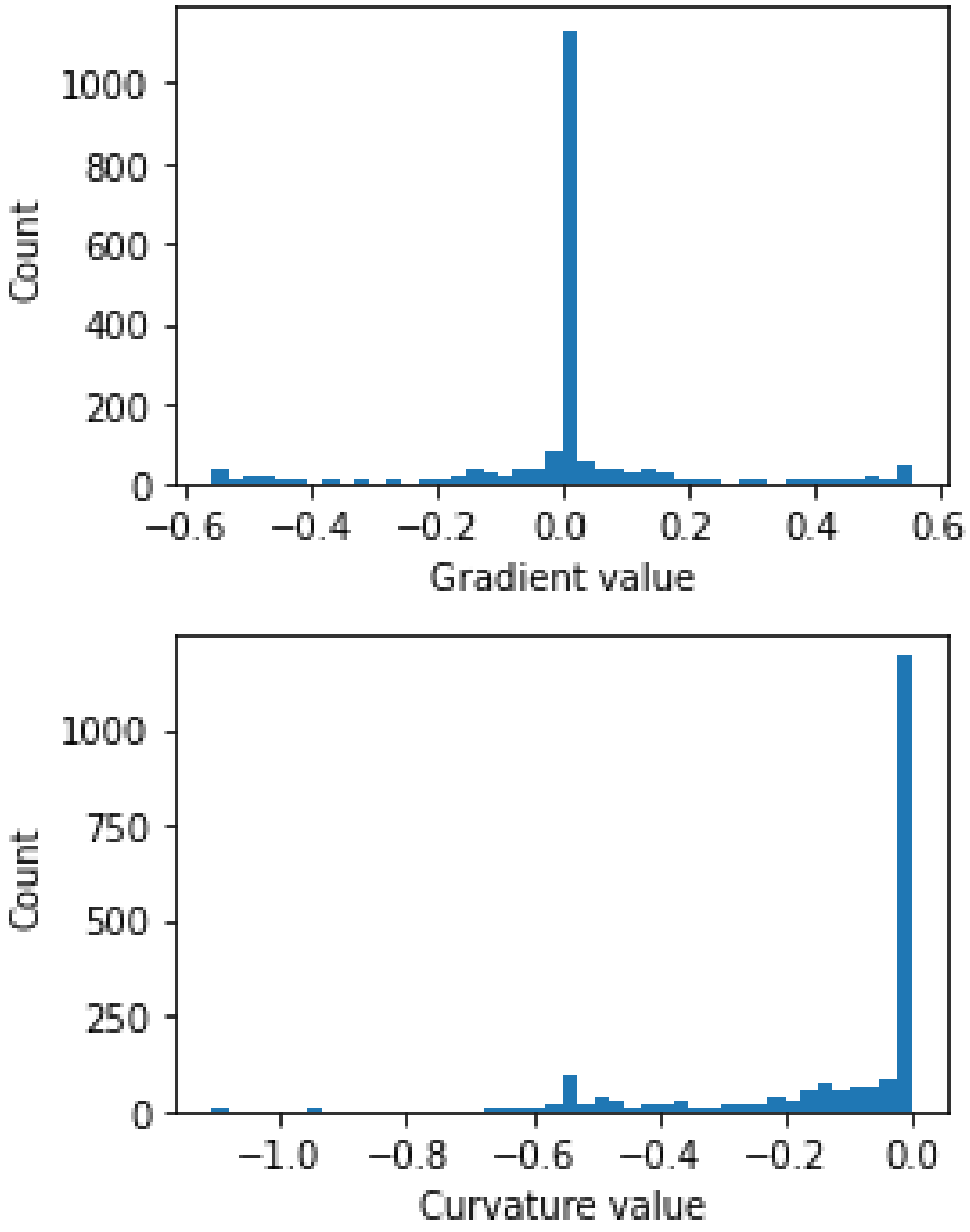


Figure 7.10: Cartpole Histogram Step 50 with baseline

7.2 LunarLander

Lunar Lander problem is a task where the solution is to properly land the lander on the landing pad. The lander slowly descends automatically and its position can be altered using the fire orientation engine. The LunarLander-v2 environment by OpenAI-Gym [16] is the discrete version of the problem. The position of the landing pad does not change and is constant at the coordinates (0,0). The state vector's first two coordinates are these coordinates determining the position of the lander. There are four actions the agent can take in this environment: fire the right orientation engine, fire the left orientation engine, fire the main engine and do nothing. If the lander successfully lands on the landing pad, it gets rewards and if it does not, it loses rewards. The rewards defer depending on whether the lander lands completely within the pad or partially in the pad. For example, each leg coming in contact with the ground is +10 reward. Similarly, the rewards for firing the left and right engines are -0.03 and firing the main engine is -0.3. An episode ends when the lander lands or crashes and this results in additional rewards between -100 and 100. The reward for solving the problem is 200.

7.2.1 Code implementation related to LunarLander

We also used the same implementation [18] of Reinforce for the LunarLander environment. This implementation includes the algorithm with and without baseline, making it easier to see the difference between the usage of using and not using a baseline. The implementation with and without baseline does not differ too much in code. For baseline, another neural network is created for the value function and the baseline loss is calculated in the Agent class, under the 'train' method. Additionally, the actor loss and value function loss is also calculated there. The need for this is explained in section 5, "Adding a Baseline". Due to the complexity of the LunarLander task being much higher than that of CartPole, it has a greater time complexity to be solved as well. Given this, with LunarLander, we are also looking at the episode length to check how adding baseline affects the episode length. There is also a condition to end the test early given the average results from the past 10 episodes is greater than 190. This will also help us investigate how fast the algorithm

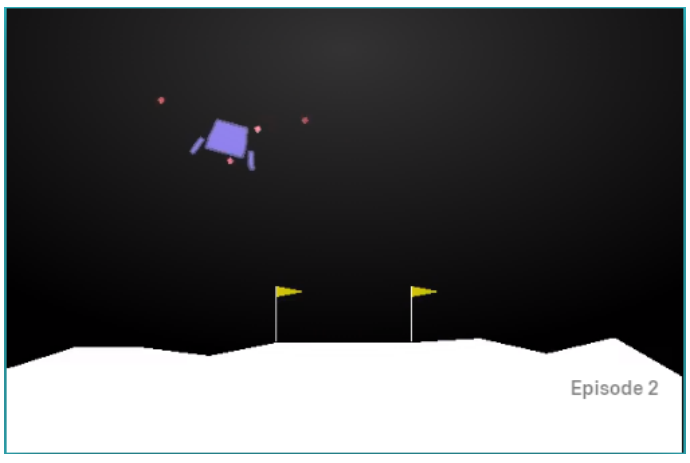


Figure 7.11: A render of LunarLander

converges with and without a baseline.

7.2.2 Results through LunarLander

For our experiments with LunarLander, we went with the following parameters: episodes = 2000, actor learning rate = 0.002, value function learning rate = 0.002. We ran the test 5 times with different seeds. The orange lines represent the data-points plotted and the blue lines represent a smoothed running average over 10 episodes.

From the graphs, we can see that they are mostly similar. However, there's one iteration that ended early at 1653 number episode. On the other hand, most tests with baseline included ended significantly early with there being an exception. One of these ended at around only 500 episodes. The average stopping episode for tests with baseline comes at around 1000 episodes. Another significant difference between the tests with and without baseline is in the episode length. We can see that for tests without baseline, the average episode length is vastly higher than the tests with baseline. Each episode takes much less time to finish with a baseline. This shows how adding a baseline can greatly help an algorithm reach convergence early and with less time taken.

7.2.3 Observations for landscape analysis for Lunar Lander:

For lunar lander we take 100 samples around the optimal parameters for the baseline implementation and the one without baseline.
In diagrams 7.16and 7.17 the points are scattered all over the place for baseline implementation yet interestingly enough it is concentrated more towards transition from minima to plateau for one without baseline at step size = 1. This might indicate that in the case of reinforce without baseline the optima reached by the learning algorithm is at a non-smooth saddle point. For step size = 10 to step size = 30 both implementations seem, the nature of the surrounding curve at those points relative to the optima seem to have smoothen towards a saddle point the points on the scatterplot approach $(0, 0)$ $\Delta_\beta^{J+} < 0$ and $\Delta_\beta^{J-} > 0$ or $(0, 0)$ $\Delta_\beta^{J+} > 0$ and $\Delta_\beta^{J-} < 0$. Points for step size = 40 and 50 seem to get scattered over $\Delta_\beta^{J+} < 0$ and $\Delta_\beta^{J-} < 0$ indicating a highly positioned valley of the aforementioned conditioned points relative to optima for no baseline. This indicates a sudden drop in curvature around those points, yet however even at step size 40 and 50 the points relative to the optima seem to retain a smooth saddle overall. Gradient and curvature analysis in 7.18 to 7.27 reinforces our observations as overall, the curvature around the optimal point reaches for 0 over increasing step sizes for baseline added gradient update but the one without any baseline implemented does not.

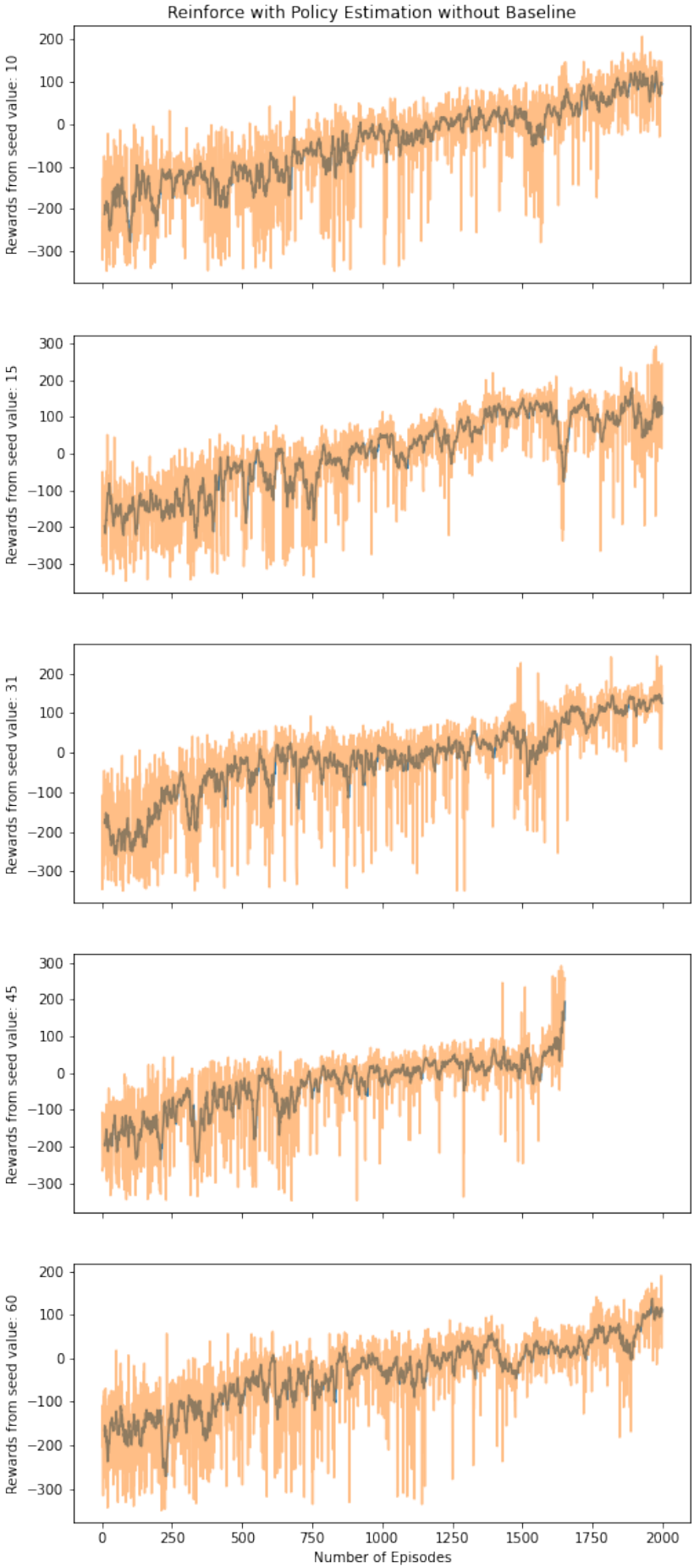


Figure 7.12: LunarLander results for tests without baseline.

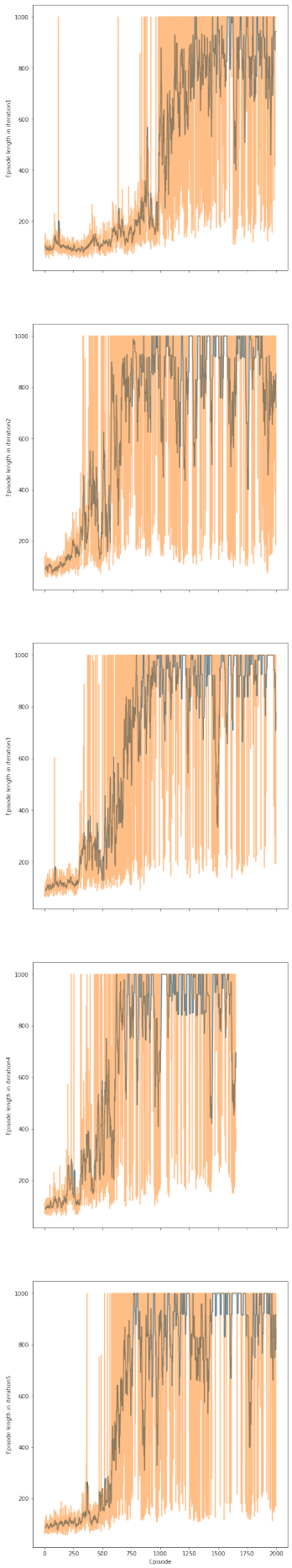


Figure 7.13: LunarLander episode length results for tests without baseline.

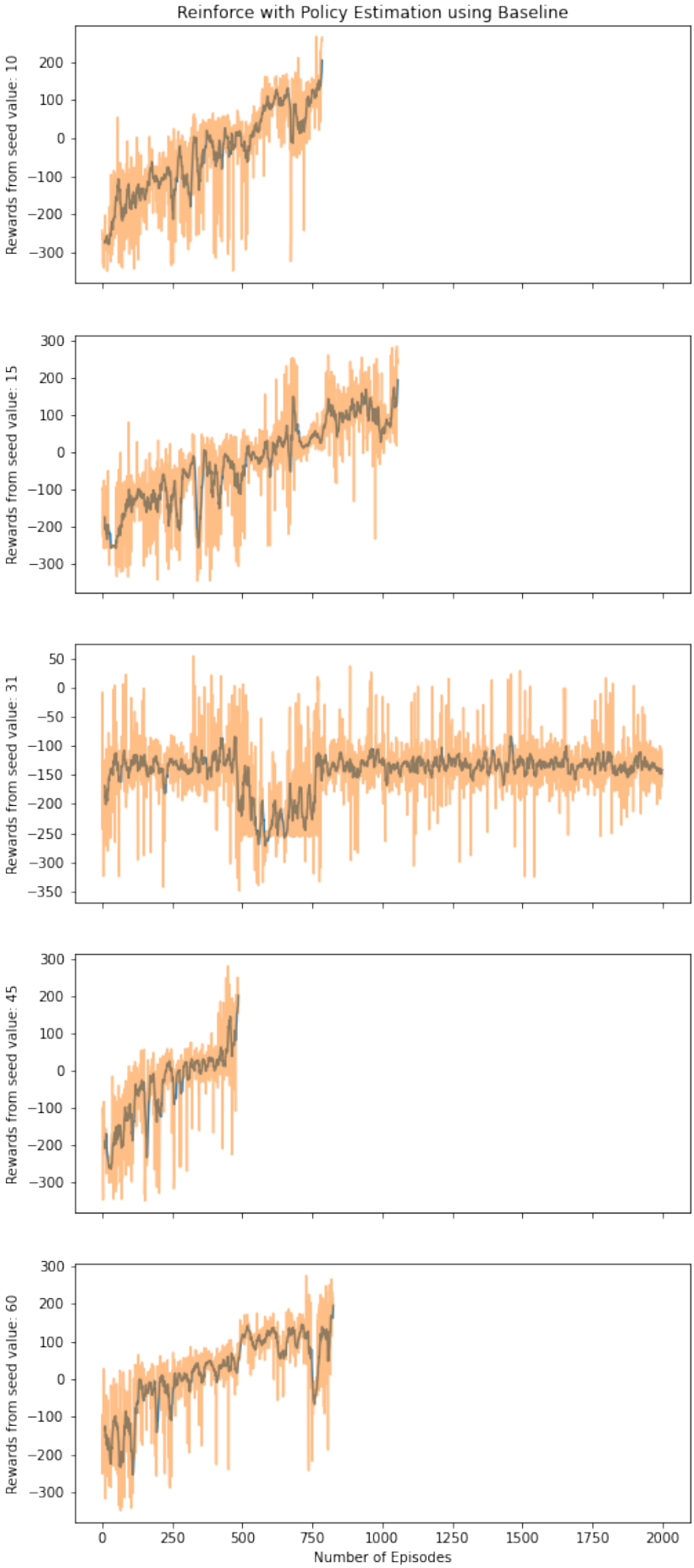


Figure 7.14: LunarLander results for tests with baseline.

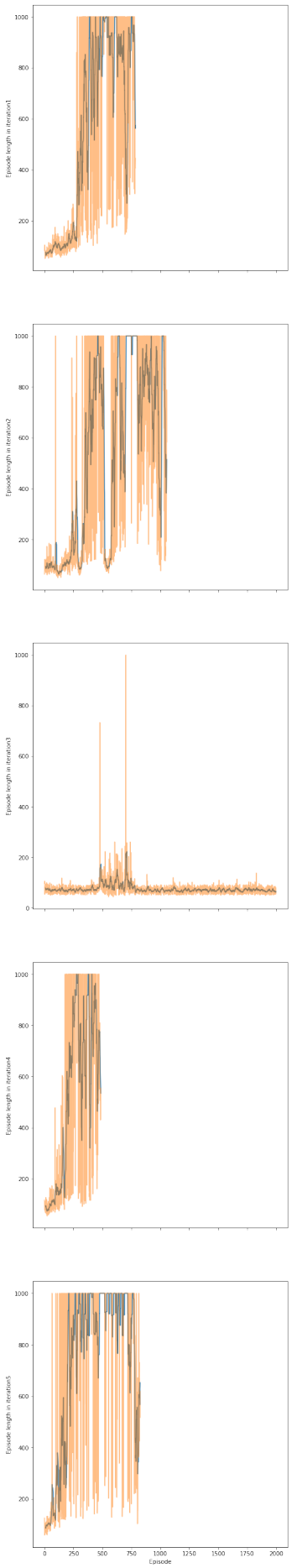


Figure 7.15: LunarLander episode length results for tests with baseline.

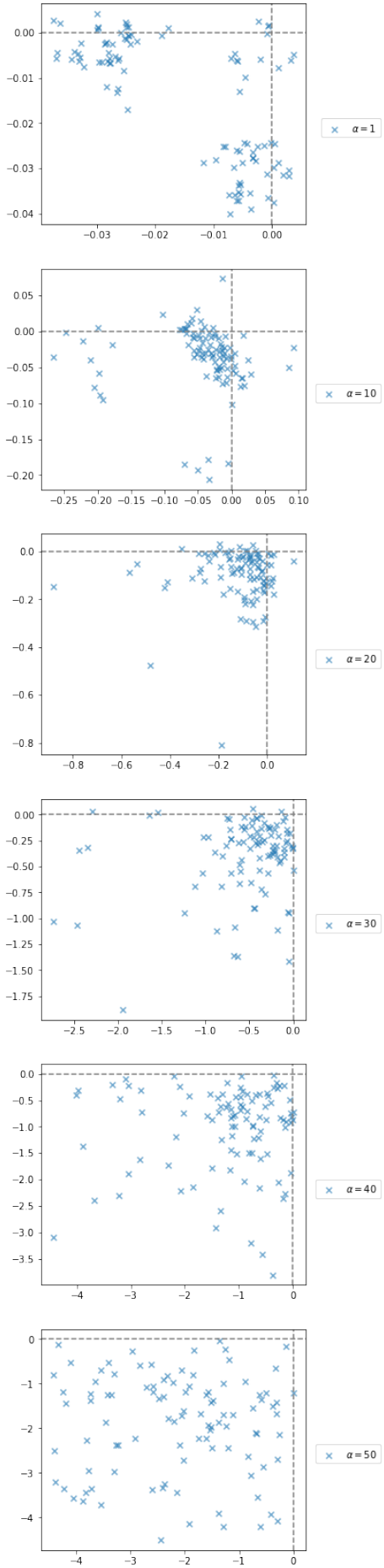


Figure 7.16: LunarLander Landscape without baseline (x-axis: Δ_β^{J+}, y-axis: Δ_β^{J-})

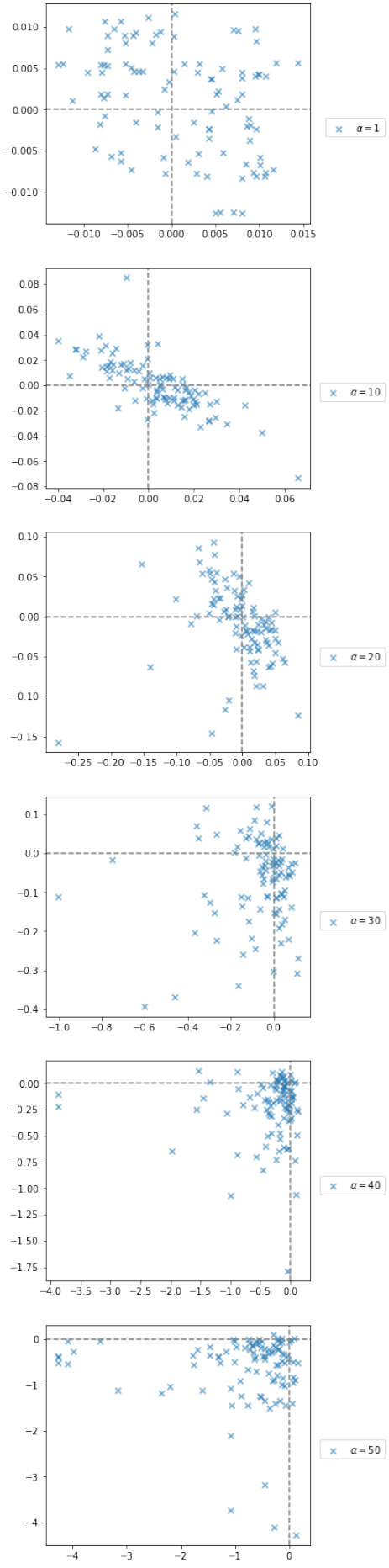


Figure 7.17: LunarLander Landscape with baseline (x-axis: Δ_β^{J+}, y-axis: Δ_β^{J-})

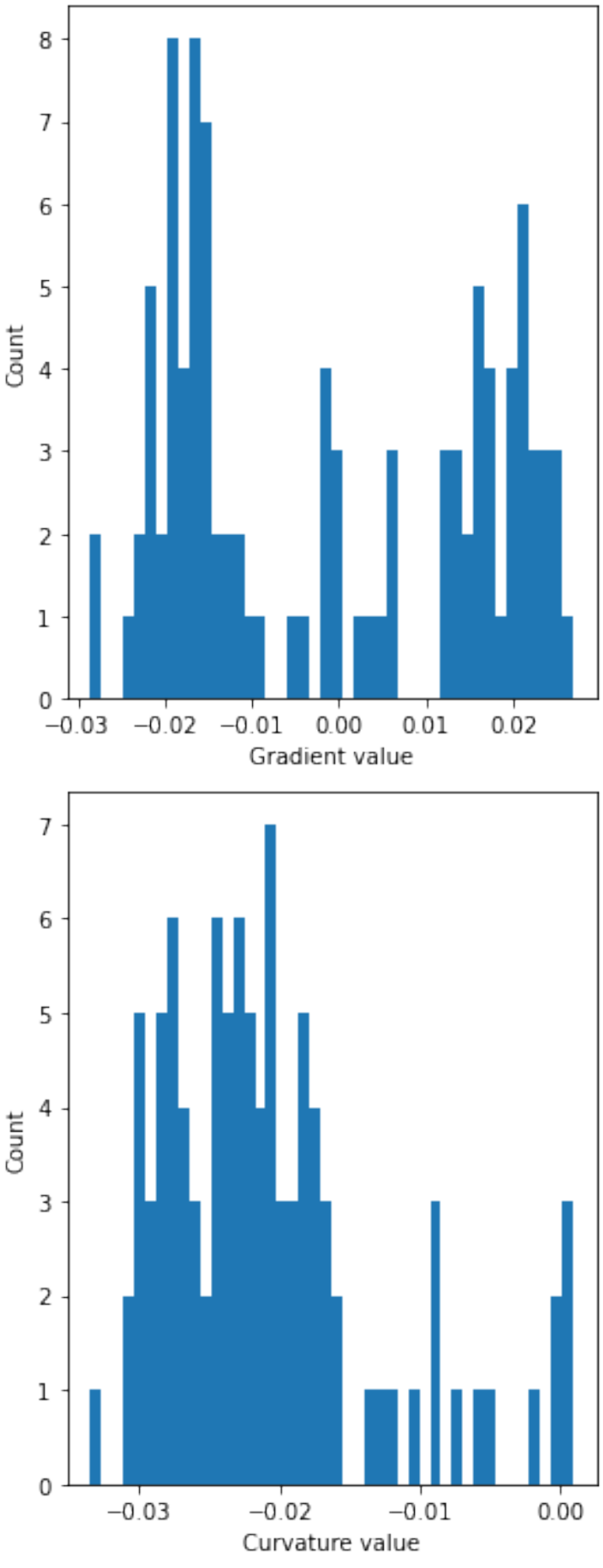


Figure 7.18: LunarLander Landscape without Baseline Histogram Step 1

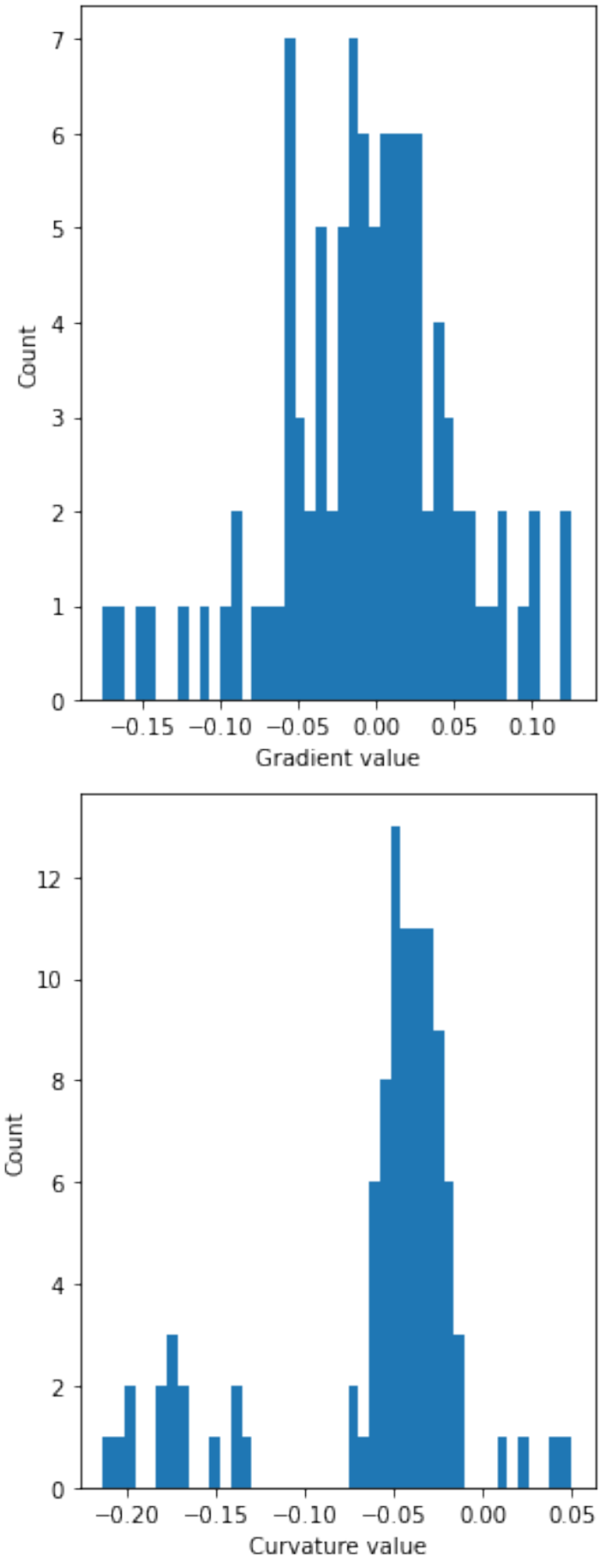


Figure 7.19: LunarLander Landscape without Baseline Histogram Step 10

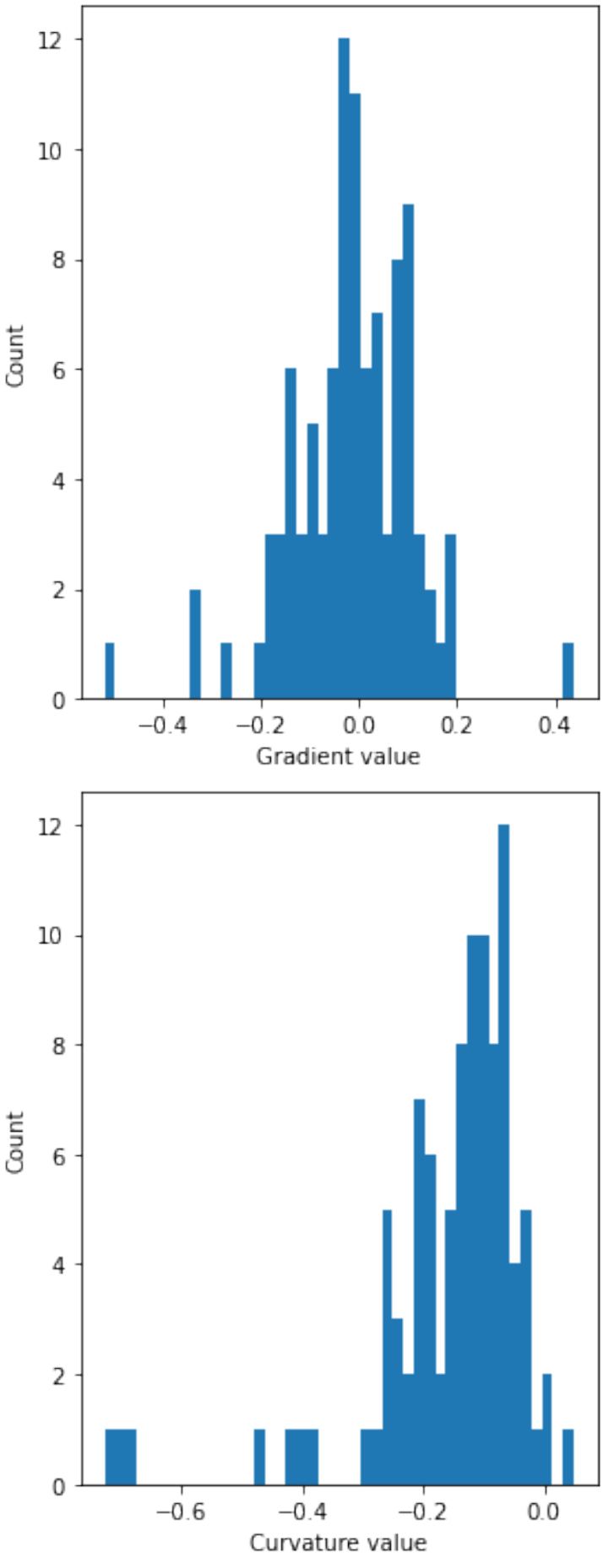


Figure 7.20: LunarLander Landscape without Baseline Histogram Step 20

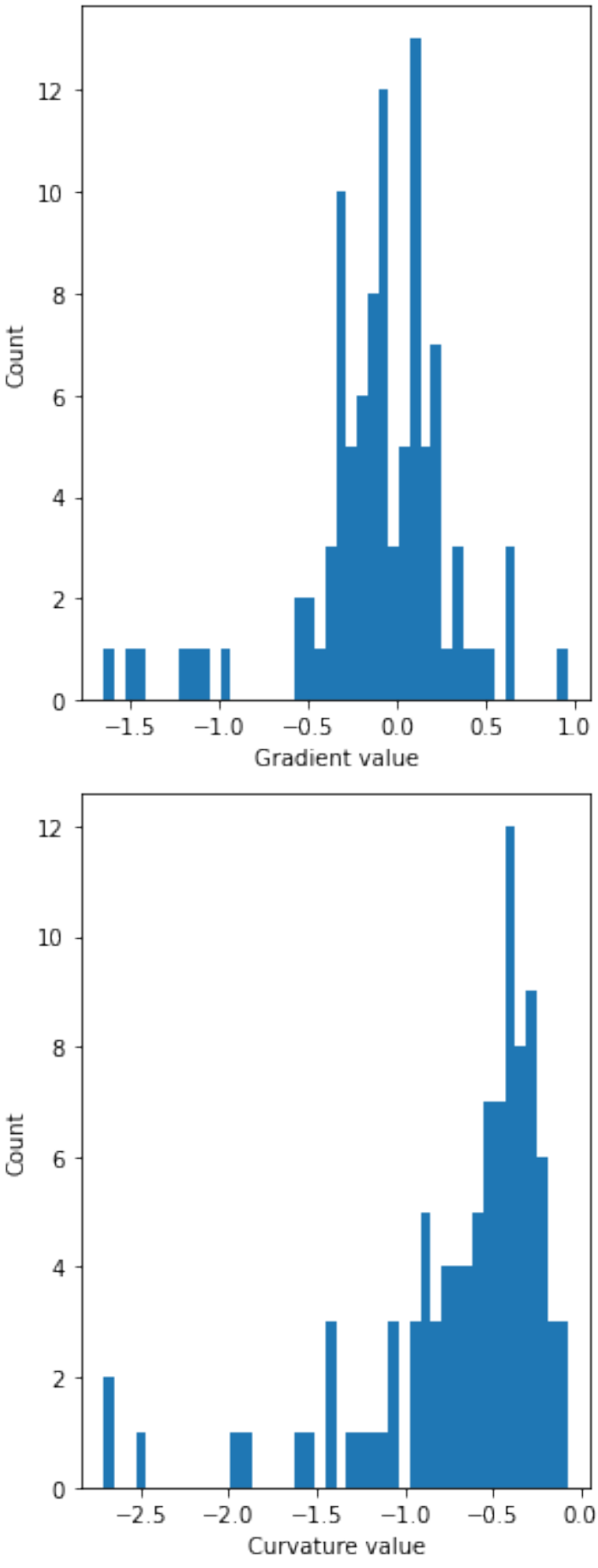


Figure 7.21: LunarLander Landscape without Baseline Histogram Step 30

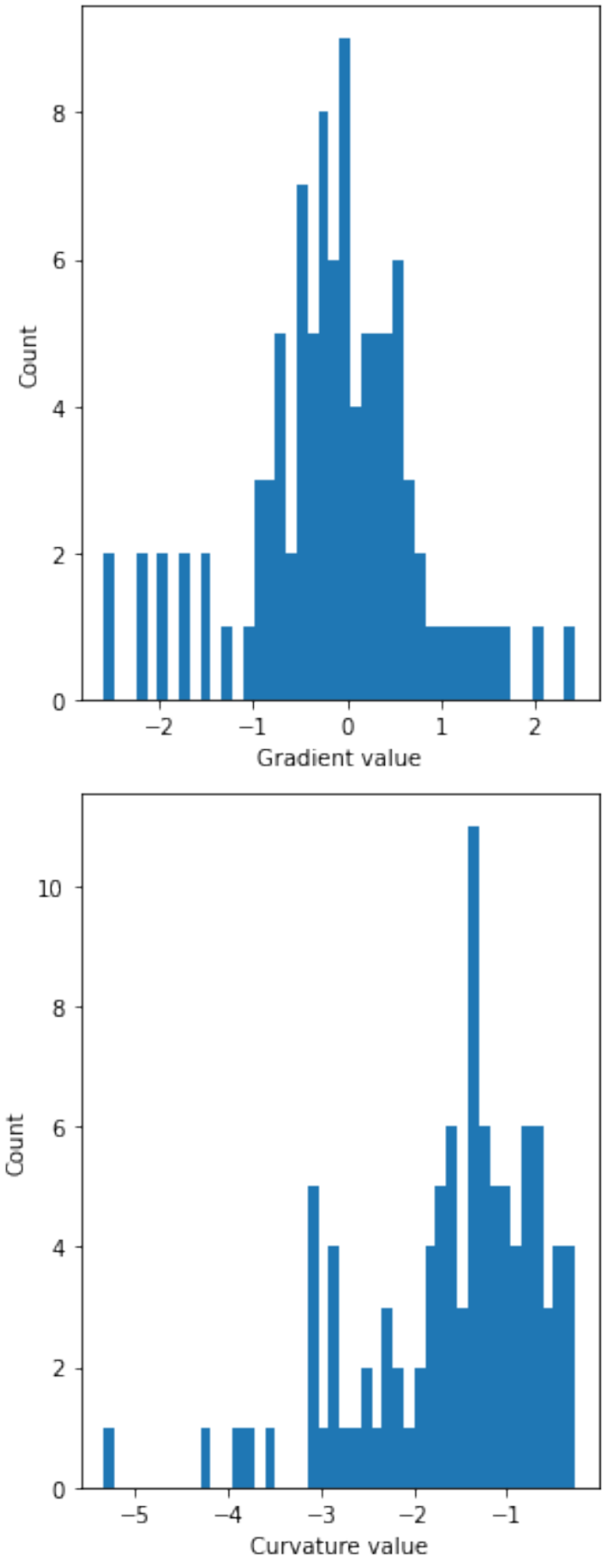


Figure 7.22: LunarLander Landscape without Baseline Histogram Step 40

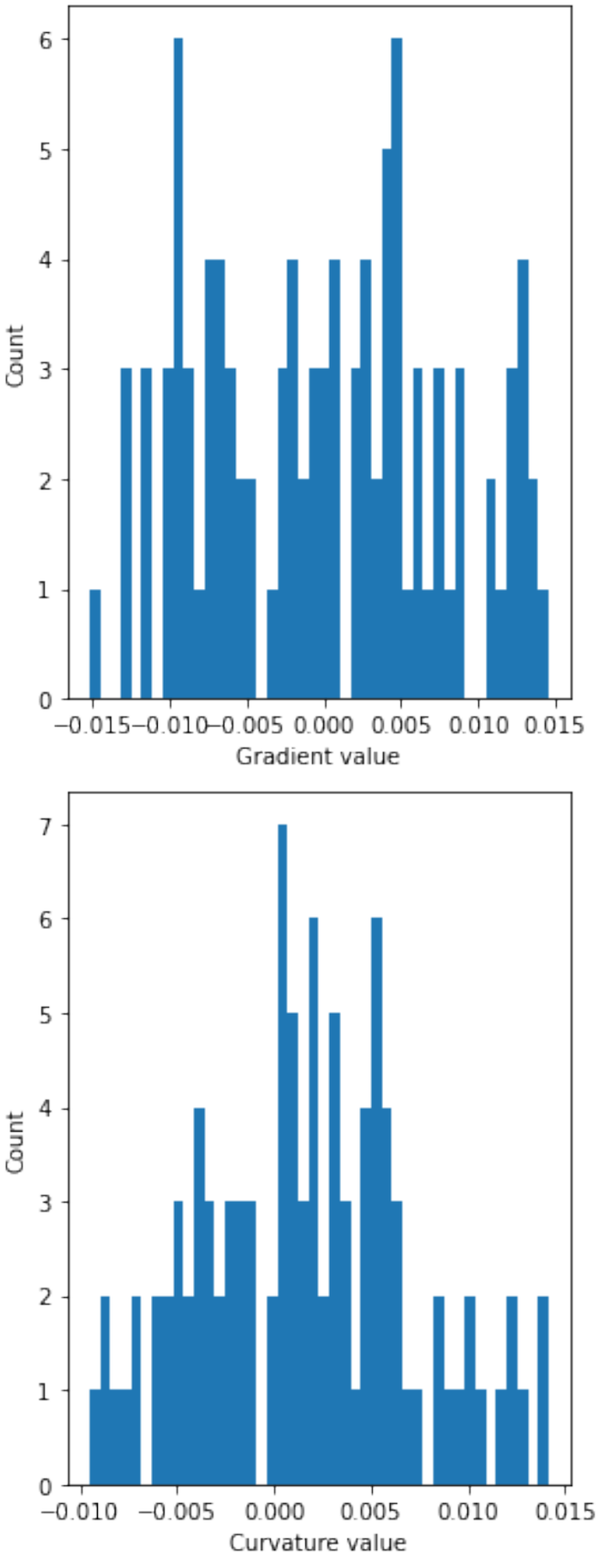


Figure 7.23: LunarLander Landscape with Baseline Histogram Step 1

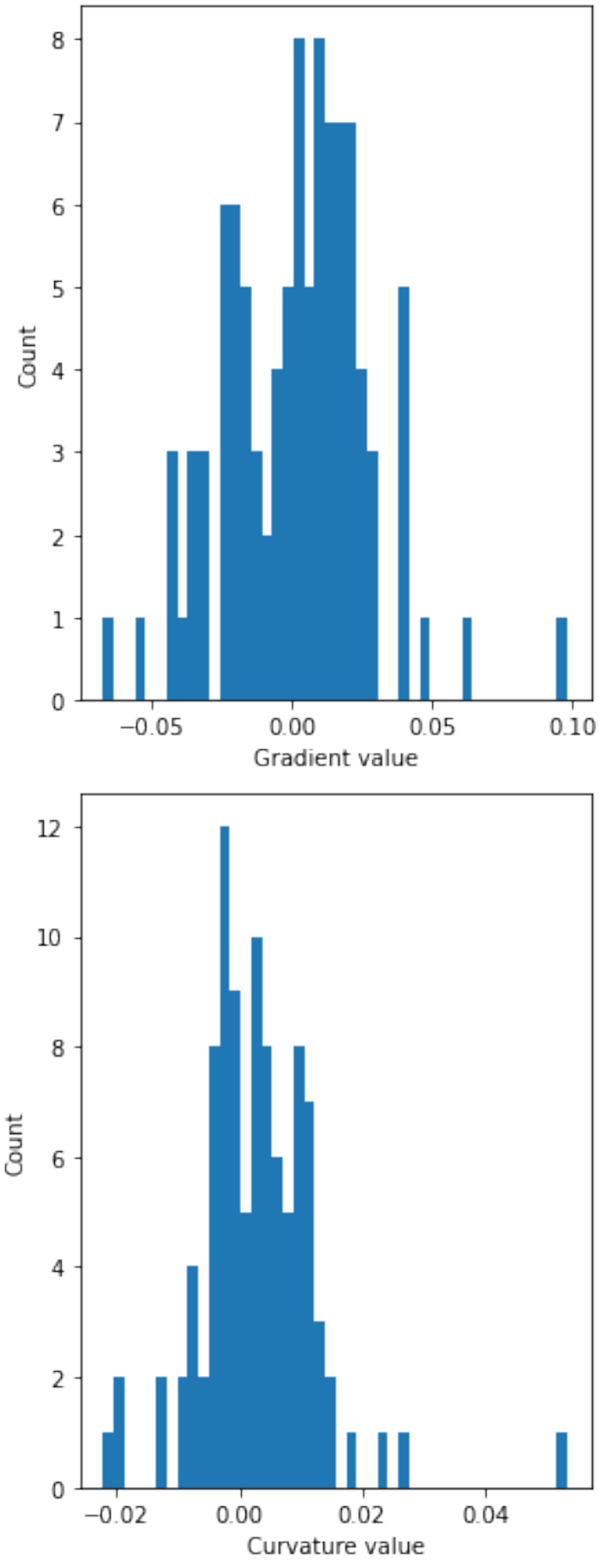


Figure 7.24: LunarLander Landscape with Baseline Histogram Step 10

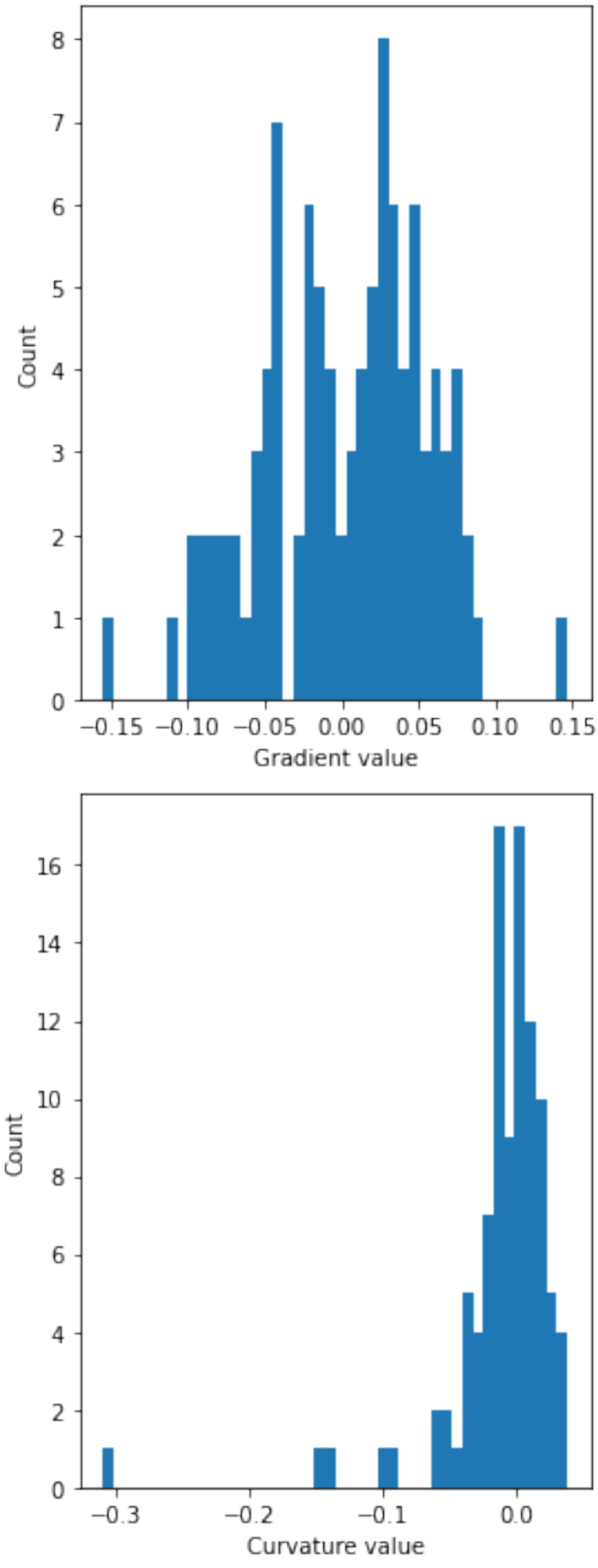


Figure 7.25: LunarLander Landscape with Baseline Histogram Step 20

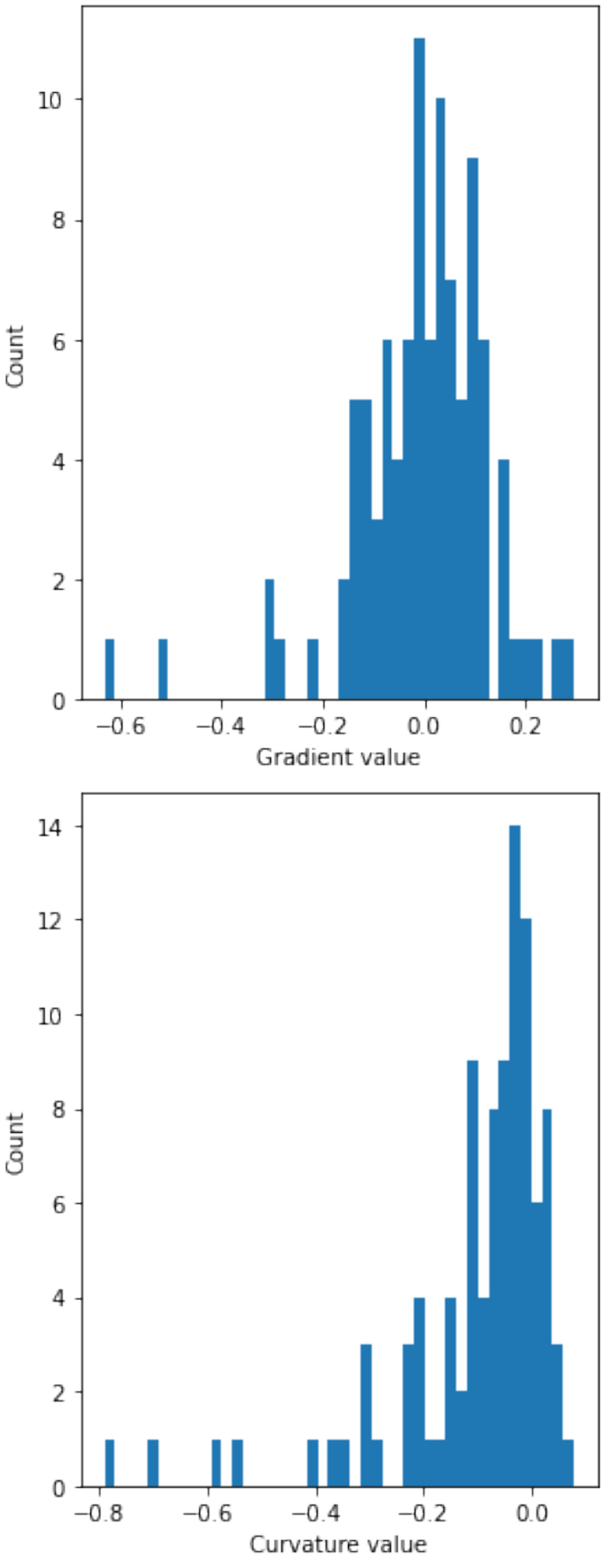


Figure 7.26: LunarLander Landscape with Baseline Histogram Step 30

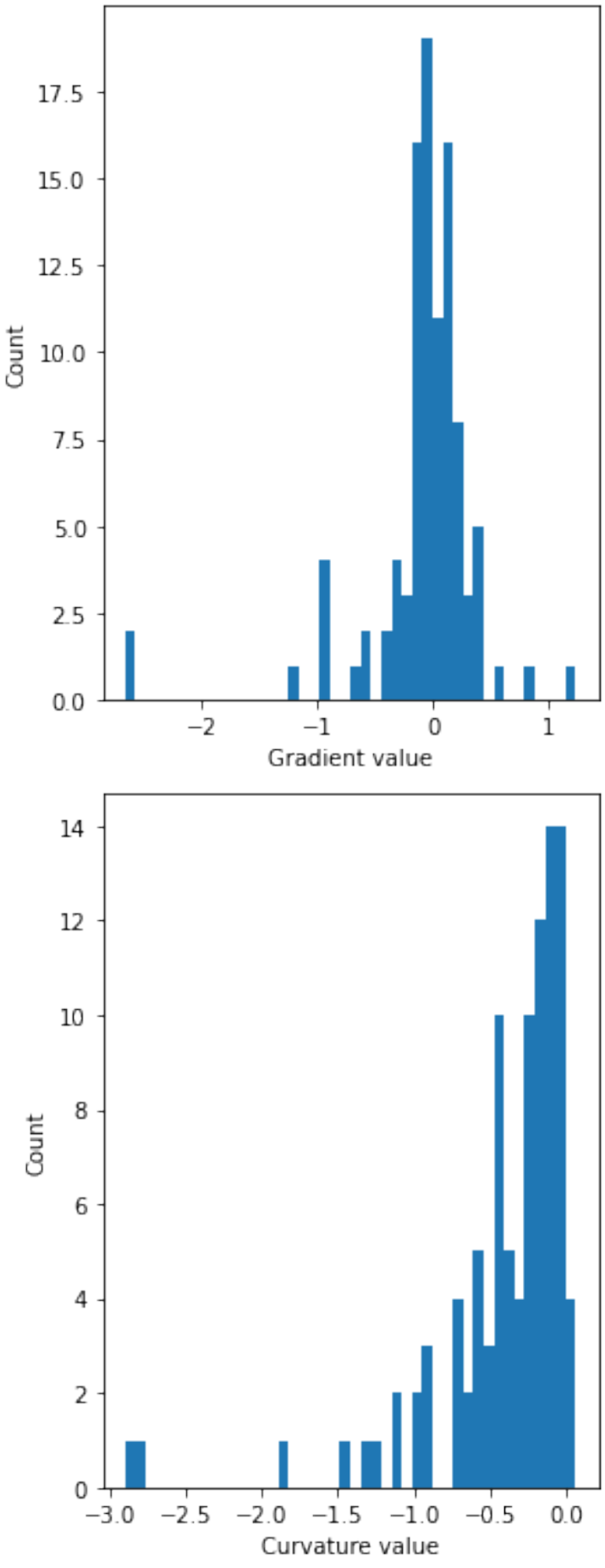


Figure 7.27: LunarLander Landscape with Baseline Histogram Step 40

Chapter 8

Conclusion and Future works

The progress in Reinforcement Learning experiments have been significant over the years. We have seen several works reconsidering already well established choices and algorithms and producing great results [9], [10]. Our work will dive deep into control variates and baselines and investigate how they affect the experiments. An issue already solved by the implementation of control variates is high variance. One such implementation of control variates are baselines [4], [8], [12]. Optimization Landscape of Reinforcement Learning algorithms is another issue to tackle when it comes to reaching optimal solution of the problem, hence policy optimization. Our aim is to find out the overall degree of effect and the nature of it on Reinforcement Learning problems by applying it using the REINFORCE [8] algorithm and also try and find out if it applies any degree of significant change to the optimization landscape of the learning algorithm. The novelty that our research aims to provide is to shed new light onto the affects of baselines and variance reduction techniques in the field of reinforcement learning by analyzing the change in the objective landscape generated by the learning algorithm with and without the implementations of these variance reduction techniques. From our experiments and the results of the landscape analysis done in chapter 7 we have observed that baselines do prove some length of an effect onto the Objective function landscape generated during learning., but the nature of it isn't the same. Perhaps the nature of the environment and the policy gradient algorithm used in tandem also play an effect. What is confirmed however that overall the landscape does experience a smoothing effect due to baselines and that might be a contributing factor to faster learning and reaching optimal or potentially suboptimal policy for a policy gradient algorithm. We chose a simple reinforcement learning tasks with a simple Actor Critic algorithm based on Reinforce with an adaptive state value baseline based on research in [14] to be able to understand the change on a more simpler scale. Going further we will also analyze the effect on landscapes of the learning agent's objective function by observing for learning methods such as TRPO and PPO and also State and Action dependent value baselines using other MDPs such as Gridworld, Atari games and MUJOCO tasks.

Bibliography

[1] S. Sharma, *The ultimate beginner's guide to reinforcement learning*, Apr. 2020. [Online]. Available: https://towardsdatascience.com/the-ultimate-beginners-guide-to-reinforcement-learning-588c071af1ec.

[2] T. Borogovac and P. Vakili, "Control variate technique: A constructive approach," Dec. 2008, pp. 320–327. DOI: 10.1109/WSC.2008.4736084.

[3] F. Ahmed, *Baseline for policy gradients that all deep learning enthusists must know*, Nov. 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/11/baseline-for-policy-gradients/.

[4] L. Weaver and N. Tao, "The optimal reward baseline for gradient-based reinforcement learning," *CoRR*, vol. abs/1301.2315, 2013. arXiv: 1301.2315. [Online]. Available: http://arxiv.org/abs/1301.2315.

[5] D. Silver, *Lectures on reinforcement learning*, Jan. 2015. [Online]. Available: https://www.davidsilver.uk/teaching/.

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. arXiv: 1707.06347. [Online]. Available: http://arxiv.org/abs/1707.06347.

[7] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *CoRR*, vol. abs/1502.05477, 2015. arXiv: 1502.05477. [Online]. Available: http://arxiv.org/abs/1502.05477.

[8] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, 1992. DOI: 10.1007/bf00992696.

[9] C. C. Hsu, C. Mendler-Dünner, and M. Hardt, "Revisiting design choices in proximal policy optimization," *CoRR*, vol. abs/2009.10897, 2020. arXiv: 2009.10897. [Online]. Available: https://arxiv.org/abs/2009.10897.

[10] L. Engstrom, A. Ilyas, S. Santurkar, *et al.*, "Implementation matters in deep policy gradients: A case study on PPO and TRPO," *CoRR*, vol. abs/2005.12729, 2020. arXiv: 2005.12729. [Online]. Available: https://arxiv.org/abs/2005.12729.

[11] A. B. Owen, *Monte Carlo theory, methods and examples.* 2013.

[12] G. Tucker, S. Bhupatiraju, S. Gu, R. E. Turner, Z. Ghahramani, and S. Levine, "The mirage of action-dependent baselines in reinforcement learning," *CoRR*, vol. abs/1802.10031, 2018. arXiv: 1802.10031. [Online]. Available: http://arxiv.org/abs/1802.10031.

[13] W. Chung, V. Thomas, M. C. Machado, and N. L. Roux, "Beyond variance reduction: Understanding the true impact of baselines on policy optimization," *CoRR*, vol. abs/2008.13773, 2020. arXiv: 2008.13773. [Online]. Available: https://arxiv.org/abs/2008.13773.

[14] Z. Ahmed, N. Le Roux, M. Norouzi, and D. Schuurmans, "Understanding the impact of entropy on policy optimization," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Sep. 2019, pp. 151–160. [Online]. Available: https://proceedings.mlr.press/v97/ahmed19a.html.

[15] G. Brockman, V. Cheung, L. Pettersson, *et al.*, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016. arXiv: 1606.01540. [Online]. Available: http://arxiv.org/abs/1606.01540.

[16] D. Takeshi, May 2017. [Online]. Available: https://danieltakeshi.github.io/2017/03/28/going-deeper-into-reinforcement-learning-fundamentals-of-policy-gradients/.

[17] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," Jun. 2015.

[18] PacktPublishing, *Packtpublishing/hands-on-reinforcement-learning-with-pytorch: Hands-on reinforcement learning with pytorch, published by [packt]*. [Online]. Available: https://github.com/PacktPublishing/Hands-on-Reinforcement-Learning-with-PyTorch.

[19] OpenAI, *The cartpole environment*. [Online]. Available: https://gym.openai.com/envs/CartPole-v0/.

[20] ——, *Lunarlander-v2 environment*. [Online]. Available: https://gym.openai.com/envs/LunarLander-v2/.

[21] M. Sahay, *Neural networks and the universal approximation theorem*, Mar. 2021. [Online]. Available: https://towardsdatascience.com/neural-networks-and-the-universal-approximation-theorem-8a389a33d30a.